《C/C++学习指南》

第23.2讲：虚拟继承,virtual的用法

作者：邵发　　QQ群：417024631
官网：http://www.afanihao.cn/
习题：http://www.afanihao.cn/kbase/

函数的重写

子类可以重写从父类继承而来的函数 （overwriting）

```
class Parent
{
public:
	void Test();
};
class Child : public Parent
{
public:
	void Test();
};
```

函数的重写

则
Child ch;
ch.Test(); // 调用的是子类的Test()函数

函数的重写

如果重写的时候，还是要嵌入调用一下父类的函数，怎么办？

```
void Child::Test()
{
    Parent::Test(); // 显式地调用父类的函数
}
```

父类指针指向子类对象

可以将父类指针指向一个子类的对象，这是完全允许的。

例如，

```
// 左侧为Tree*，右侧为AppleTree*
Tree*  p = new AppleTree();
```

从普通的逻辑来讲，苹果树是一种树，因而可以把AppleTree*视为一种Tree*

从语法本质上讲，子类对象的前半部分就是父类，因而可以将子类对象的指针直接转化为父类。

父类指针指向子类对象

有父类和子类:

```
class Parent
{
public:
      int a;
};

class Child : public Parent
{
public:
      int b;
};
```

父类指针指向子类对象

```
int main()
{
	Child ch;
	ch.a = 0x11111111;
	ch.b = 0x22222222;

	Parent* p = &ch; // p指向的对象是Child*
	printf("Parent::a = %d \n", p->a);

	return 0;
}
```

所以，从直观感觉到内在机理都允许这么做

父类指针指向子类对象

问题：考虑以下情况，

```
Parent* p = new Child();
p->Test();
```

那么，此时调用的Test()是父类的、还是子类的？

(1)指针p的类型是Parent*

(2)指针p指向的对象是Child*

调用者的初衷：因为p指向的是对象是子类对象，所以应该调用子类的Test()。

虚拟继承：virtual

当一个成员函数需要子类重写，那么在父类应该将其声明为virtual。
（有时将声明为virtual的函数为“虚函数”）

例如

```
class Parent
{
public:
	virtual void Test();
};
```

virtual本身表明该函数即将被子类重写。

虚拟继承：virtual

加virtual关键字是必要的。

考虑以下情况，

```
Parent* obj = new Child(); // 语法允许，合乎情理
obj->Test();
```

此时，如果Test()在父类中被声明为virtual，是调用的是子类的Test()。

这解释了virtual的作用：根据对象的实际类型，调用相应类型的函数。

虚拟继承: virtual

注意：
(1)只需要在父类中将函数声明为virtual，子类自动地就是virtual了。

(2)即将被重写的函数添加virtual，是一条应该遵守的编码习惯。

小结

1. 介绍继承关系中，对函数重写后的结果
2. 介绍virtual关键字的作用和必要性（父类指针指向子类对象）